face-emotion-recongition

December 23, 2023

[400]:

```
import tensorflow as tf

import numpy as np
import matplotlib.pyplot as plt
import cv2
import shutil
import os

from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras import regularizers
from keras.layers import Dense, MaxPooling2D, Flatten, Conv2D, Dropout ,␣
 ↪BatchNormalization, Activation
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from keras.optimizers import Adam, SGD
from keras.utils import to_categorical
from sklearn.utils.class_weight import compute_class_weight
from imblearn.over_sampling import SMOTE
```

[401]:

```
# # Define the path to your original dataset and the paths where you want to␣
 ↪store your train and test datasets
# original_dataset_dir = 'CK+'
# train_dir = 'CK+train'
# validate_dir = 'CK+validate'
# test_dir = 'CK+test'

# # Create directories for training and testing datasets if they do not exist
# os.makedirs(train_dir, exist_ok=True)
# os.makedirs(validate_dir, exist_ok=True)
# os.makedirs(test_dir, exist_ok=True)

# # Define the split ratio
# train_ratio = 0.7

# # Loop through each emotion category in the original dataset
# for emotion in os.listdir(original_dataset_dir):
#     emotion_dir = os.path.join(original_dataset_dir, emotion)
```

```
#     if os.path.isdir(emotion_dir):
#         # Get a list of all the image filenames in the emotion category
#         images = [f for f in os.listdir(emotion_dir) if os.path.isfile(os.
↪path.join(emotion_dir, f))]

#         # Randomly shuffle the list of image filenames
#         np.random.shuffle(images)

#         # Split the list of image filenames into training and testing sets
#         train_size = int(len(images) * train_ratio)
#         validate_size = int(len(images)-len(images)*(15/100))
#         train_images = images[:train_size]
#         validate_images = images[train_size:validate_size]
#         test_images = images[validate_size:]

#         # Create directories for the emotion category in the train and test␣
↪datasets
#         train_emotion_dir = os.path.join(train_dir, emotion)
#         validate_emotion_dir = os.path.join(validate_dir, emotion)
#         test_emotion_dir = os.path.join(test_dir, emotion)
#         os.makedirs(train_emotion_dir, exist_ok=True)
#         os.makedirs(validate_emotion_dir, exist_ok=True)
#         os.makedirs(test_emotion_dir, exist_ok=True)

#         # Copy the images into the corresponding directories
#         for image in train_images:
#             shutil.copy(os.path.join(emotion_dir, image), os.path.
↪join(train_emotion_dir, image))
#         for image in validate_images:
#             shutil.copy(os.path.join(emotion_dir, image), os.path.
↪join(validate_emotion_dir, image))
#         for image in test_images:
#             shutil.copy(os.path.join(emotion_dir, image), os.path.
↪join(test_emotion_dir, image))

# print("Dataset splitting complete")
```

[402]:
```
# Create a data generator with augmentation
trainDataGenerator = ImageDataGenerator(
    rescale=1./255,  # Rescale the pixel values (normalization)
    width_shift_range=0.1,  # Random horizontal shifts (15% of total width)
    height_shift_range=0.1,  # Random vertical shifts (15% of total height)
    horizontal_flip=True,  # Randomly flip inputs horizontally
    fill_mode='nearest',
    zoom_range=0.1,
    shear_range=0.1,
)
```

```
# Load images from the directory and apply the defined transformations
trainingData = trainDataGenerator.flow_from_directory(
    'CK+train',  # Path to the training data
    target_size=(48, 48),  # Resize images to 48x48
    class_mode='categorical',  # Labels will be returned in categorical format
    batch_size=8
)



class_labels = list(trainingData.class_indices.keys())
class_counts = np.zeros(len(class_labels))
class_weights = compute_class_weight(class_weight = 'balanced', classes = np.
 ↪unique(trainingData.labels), y = trainingData.labels)

class_weight_dict = {class_idx: weight for class_idx, weight in␣
 ↪enumerate(class_weights)}

#smote = SMOTE(sampling_strategy='auto', random_state=42)

#trainingData[0][0], trainingData[0][1] = smote.
 ↪fit_resample(trainingData[0][0], np.argmax(trainingData[0][1], axis=1))

print(class_weight_dict)
```

```
Found 682 images belonging to 7 classes.
{0: 1.0364741641337385, 1: 2.6332046332046333, 2: 0.7921022067363531, 3:
1.8736263736263736, 4: 0.6765873015873016, 5: 1.6798029556650247, 6:
0.5599343185550082}
```

[403]:

```
# Initialize an ImageDataGenerator for test data with rescaling
validationDataGenerator = ImageDataGenerator(rescale=1./255,  # Rescale the␣
 ↪pixel values (normalization)
)
# Creates a data generator for the test dataset
# flow_from_directory method loads images from a directory
validationData = validationDataGenerator.flow_from_directory(
    'CK+validate',  # Directory path for test images
    target_size = (48, 48),  # Resizes images to 48x48 pixels
    class_mode = 'categorical',  # Images are classified categorically
    batch_size=8
)

# validationData is now a generator that yields batches of test images and␣
 ↪their labels
validationData.class_indices
```

```
Found 147 images belonging to 7 classes.
```

[403]:
```
{'anger': 0,
 'contempt': 1,
 'disgust': 2,
 'fear': 3,
 'happy': 4,
 'sadness': 5,
 'surprise': 6}
```

[404]:
```
testDataGenerator = ImageDataGenerator(rescale=1./255)

testingData = testDataGenerator.flow_from_directory(
    'CK+test',  # Directory path for test images
    target_size = (48, 48),  # Resizes images to 48x48 pixels
    class_mode = 'categorical',  # Images are classified categorically
    batch_size=8,
    shuffle = False,
)

testDataGenerator2 = ImageDataGenerator(rescale=1./255)

testingData2 = testDataGenerator2.flow_from_directory(
    'data/test',  # Directory path for test images
    target_size = (48, 48),  # Resizes images to 48x48 pixels
    class_mode = 'categorical',  # Images are classified categorically
    batch_size=8,
    shuffle = False,
)

testingData.class_indices
```

```
Found 152 images belonging to 7 classes.
Found 3589 images belonging to 7 classes.
```

[404]:
```
{'anger': 0,
 'contempt': 1,
 'disgust': 2,
 'fear': 3,
 'happy': 4,
 'sadness': 5,
 'surprise': 6}
```

[405]:
```
model = Sequential()

model.
 ↪add(Conv2D(32,(3,3),padding="same",input_shape=(48,48,3),activation='relu'))
```

```python
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())

model.add(Dense(128))
model.add(Dense(64))
model.add(Dense(32))
model.add(Dropout(0.25))

model.add(Dense(7,activation='softmax'))

opts = SGD(
    learning_rate=0.01, nesterov=True
)
model.
 ↪compile(optimizer=opts,loss='categorical_crossentropy',metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_37"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_184 (Conv2D)         (None, 48, 48, 32)        896

 conv2d_185 (Conv2D)         (None, 46, 46, 32)        9248

 max_pooling2d_100 (MaxPool  (None, 23, 23, 32)        0
 ing2D)

 conv2d_186 (Conv2D)         (None, 21, 21, 64)        18496

 conv2d_187 (Conv2D)         (None, 19, 19, 64)        36928

 max_pooling2d_101 (MaxPool  (None, 9, 9, 64)          0
 ing2D)

 conv2d_188 (Conv2D)         (None, 7, 7, 128)         73856
```

```
 conv2d_189 (Conv2D)         (None, 5, 5, 128)         147584

 max_pooling2d_102 (MaxPool  (None, 2, 2, 128)         0
 ing2D)

 flatten_37 (Flatten)        (None, 512)               0

 dense_132 (Dense)           (None, 128)               65664

 dense_133 (Dense)           (None, 64)                8256

 dense_134 (Dense)           (None, 32)                2080

 dropout_11 (Dropout)        (None, 32)                0

 dense_135 (Dense)           (None, 7)                 231

=================================================================
Total params: 363239 (1.39 MB)
Trainable params: 363239 (1.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[406]:

```
early_stopping = EarlyStopping(
    monitor='val_loss',
    min_delta=0.0005,
    patience=20,
    verbose=1,
    restore_best_weights=True,
)

lr_scheduler = ReduceLROnPlateau(
    monitor='accuracy',
    factor=0.5,
    patience=5,
    min_lr=0.0000001,
    verbose=1,
)

callbacks = [
    early_stopping,
    lr_scheduler,
]

history = model.fit(
    trainingData,
```

```
    epochs=256,
    validation_data=validationData,
    steps_per_epoch= trainingData.n//trainingData.batch_size,
    callbacks=callbacks,
    class_weight=class_weight_dict
)
model.save('FER1.keras')
```

```
Epoch 1/256
85/85 [==============================] - 2s 7ms/step - loss: 1.9563 - accuracy:
0.1543 - val_loss: 1.9593 - val_accuracy: 0.2109 - lr: 0.0100
Epoch 2/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9469 - accuracy:
0.1217 - val_loss: 1.9437 - val_accuracy: 0.1224 - lr: 0.0100
Epoch 3/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9379 - accuracy:
0.1513 - val_loss: 1.9299 - val_accuracy: 0.2177 - lr: 0.0100
Epoch 4/256
85/85 [==============================] - 0s 6ms/step - loss: 1.9547 - accuracy:
0.1691 - val_loss: 1.9538 - val_accuracy: 0.1905 - lr: 0.0100
Epoch 5/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9550 - accuracy:
0.1439 - val_loss: 1.9511 - val_accuracy: 0.0748 - lr: 0.0100
Epoch 6/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9409 - accuracy:
0.2136 - val_loss: 1.9228 - val_accuracy: 0.2109 - lr: 0.0100
Epoch 7/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9417 - accuracy:
0.1899 - val_loss: 1.9251 - val_accuracy: 0.2177 - lr: 0.0100
Epoch 8/256
85/85 [==============================] - 0s 6ms/step - loss: 1.9334 - accuracy:
0.1766 - val_loss: 1.9213 - val_accuracy: 0.2313 - lr: 0.0100
Epoch 9/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9314 - accuracy:
0.1884 - val_loss: 1.9837 - val_accuracy: 0.0544 - lr: 0.0100
Epoch 10/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9401 - accuracy:
0.1736 - val_loss: 1.8747 - val_accuracy: 0.4014 - lr: 0.0100
Epoch 11/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9349 - accuracy:
0.2181 - val_loss: 1.8325 - val_accuracy: 0.4014 - lr: 0.0100
Epoch 12/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9209 - accuracy:
0.2166 - val_loss: 1.8254 - val_accuracy: 0.4150 - lr: 0.0100
Epoch 13/256
85/85 [==============================] - 1s 6ms/step - loss: 1.9028 - accuracy:
0.2418 - val_loss: 1.7592 - val_accuracy: 0.3537 - lr: 0.0100
```

```
Epoch 14/256
85/85 [==============================] - 1s 6ms/step - loss: 1.8362 - accuracy:
0.3205 - val_loss: 1.5843 - val_accuracy: 0.5578 - lr: 0.0100
Epoch 15/256
85/85 [==============================] - 1s 6ms/step - loss: 1.7463 - accuracy:
0.4050 - val_loss: 1.4147 - val_accuracy: 0.4558 - lr: 0.0100
Epoch 16/256
85/85 [==============================] - 1s 6ms/step - loss: 1.6539 - accuracy:
0.4080 - val_loss: 1.2165 - val_accuracy: 0.5374 - lr: 0.0100
Epoch 17/256
85/85 [==============================] - 1s 6ms/step - loss: 1.5721 - accuracy:
0.4614 - val_loss: 1.1388 - val_accuracy: 0.5850 - lr: 0.0100
Epoch 18/256
85/85 [==============================] - 1s 6ms/step - loss: 1.5519 - accuracy:
0.4481 - val_loss: 1.1257 - val_accuracy: 0.5850 - lr: 0.0100
Epoch 19/256
85/85 [==============================] - 0s 6ms/step - loss: 1.4675 - accuracy:
0.5059 - val_loss: 1.1826 - val_accuracy: 0.5714 - lr: 0.0100
Epoch 20/256
85/85 [==============================] - 1s 6ms/step - loss: 1.4458 - accuracy:
0.5193 - val_loss: 1.0302 - val_accuracy: 0.6190 - lr: 0.0100
Epoch 21/256
85/85 [==============================] - 1s 6ms/step - loss: 1.3443 - accuracy:
0.5712 - val_loss: 1.1028 - val_accuracy: 0.5918 - lr: 0.0100
Epoch 22/256
85/85 [==============================] - 1s 6ms/step - loss: 1.3870 - accuracy:
0.5415 - val_loss: 0.9638 - val_accuracy: 0.6054 - lr: 0.0100
Epoch 23/256
85/85 [==============================] - 1s 6ms/step - loss: 1.2349 - accuracy:
0.6206 - val_loss: 0.9548 - val_accuracy: 0.6531 - lr: 0.0100
Epoch 24/256
85/85 [==============================] - 1s 6ms/step - loss: 1.2867 - accuracy:
0.5727 - val_loss: 0.9148 - val_accuracy: 0.6599 - lr: 0.0100
Epoch 25/256
85/85 [==============================] - 0s 6ms/step - loss: 1.2669 - accuracy:
0.6009 - val_loss: 0.9005 - val_accuracy: 0.6939 - lr: 0.0100
Epoch 26/256
85/85 [==============================] - 1s 6ms/step - loss: 1.1702 - accuracy:
0.6320 - val_loss: 1.1208 - val_accuracy: 0.6122 - lr: 0.0100
Epoch 27/256
85/85 [==============================] - 1s 6ms/step - loss: 1.2016 - accuracy:
0.6142 - val_loss: 1.0293 - val_accuracy: 0.6122 - lr: 0.0100
Epoch 28/256
85/85 [==============================] - 1s 6ms/step - loss: 1.1435 - accuracy:
0.6543 - val_loss: 0.8197 - val_accuracy: 0.7211 - lr: 0.0100
Epoch 29/256
85/85 [==============================] - 1s 6ms/step - loss: 1.1005 - accuracy:
0.6691 - val_loss: 0.7415 - val_accuracy: 0.7619 - lr: 0.0100
```

```
Epoch 30/256
85/85 [==============================] - 1s 6ms/step - loss: 1.1017 - accuracy:
0.6662 - val_loss: 0.7877 - val_accuracy: 0.7211 - lr: 0.0100
Epoch 31/256
85/85 [==============================] - 1s 6ms/step - loss: 1.0324 - accuracy:
0.6647 - val_loss: 0.9133 - val_accuracy: 0.6531 - lr: 0.0100
Epoch 32/256
85/85 [==============================] - 1s 6ms/step - loss: 0.9865 - accuracy:
0.7047 - val_loss: 0.7427 - val_accuracy: 0.7687 - lr: 0.0100
Epoch 33/256
85/85 [==============================] - 1s 6ms/step - loss: 1.0014 - accuracy:
0.6914 - val_loss: 1.0835 - val_accuracy: 0.6122 - lr: 0.0100
Epoch 34/256
85/85 [==============================] - 1s 6ms/step - loss: 0.9603 - accuracy:
0.7047 - val_loss: 0.5518 - val_accuracy: 0.8231 - lr: 0.0100
Epoch 35/256
85/85 [==============================] - 1s 6ms/step - loss: 0.9661 - accuracy:
0.7077 - val_loss: 0.7429 - val_accuracy: 0.7415 - lr: 0.0100
Epoch 36/256
85/85 [==============================] - 1s 6ms/step - loss: 0.9049 - accuracy:
0.7240 - val_loss: 0.5543 - val_accuracy: 0.7823 - lr: 0.0100
Epoch 37/256
85/85 [==============================] - 1s 6ms/step - loss: 0.8339 - accuracy:
0.7389 - val_loss: 0.7729 - val_accuracy: 0.6871 - lr: 0.0100
Epoch 38/256
85/85 [==============================] - 1s 6ms/step - loss: 0.8827 - accuracy:
0.7240 - val_loss: 0.5490 - val_accuracy: 0.8095 - lr: 0.0100
Epoch 39/256
85/85 [==============================] - 1s 6ms/step - loss: 0.8219 - accuracy:
0.7448 - val_loss: 0.6767 - val_accuracy: 0.7823 - lr: 0.0100
Epoch 40/256
85/85 [==============================] - 1s 6ms/step - loss: 0.7988 - accuracy:
0.7478 - val_loss: 0.5424 - val_accuracy: 0.8299 - lr: 0.0100
Epoch 41/256
85/85 [==============================] - 1s 6ms/step - loss: 0.8086 - accuracy:
0.7626 - val_loss: 0.5121 - val_accuracy: 0.8571 - lr: 0.0100
Epoch 42/256
85/85 [==============================] - 0s 6ms/step - loss: 0.8245 - accuracy:
0.7507 - val_loss: 0.5448 - val_accuracy: 0.8095 - lr: 0.0100
Epoch 43/256
85/85 [==============================] - 1s 6ms/step - loss: 0.7201 - accuracy:
0.7685 - val_loss: 0.4164 - val_accuracy: 0.8571 - lr: 0.0100
Epoch 44/256
85/85 [==============================] - 1s 6ms/step - loss: 0.7257 - accuracy:
0.7834 - val_loss: 0.9014 - val_accuracy: 0.6871 - lr: 0.0100
Epoch 45/256
85/85 [==============================] - 1s 6ms/step - loss: 0.7727 - accuracy:
0.7626 - val_loss: 0.4283 - val_accuracy: 0.8503 - lr: 0.0100
```

```
Epoch 46/256
85/85 [==============================] - 0s 6ms/step - loss: 0.6478 - accuracy:
0.8012 - val_loss: 0.4199 - val_accuracy: 0.8367 - lr: 0.0100
Epoch 47/256
85/85 [==============================] - 1s 6ms/step - loss: 0.6879 - accuracy:
0.7789 - val_loss: 0.3767 - val_accuracy: 0.8503 - lr: 0.0100
Epoch 48/256
85/85 [==============================] - 1s 6ms/step - loss: 0.5963 - accuracy:
0.8264 - val_loss: 0.4654 - val_accuracy: 0.8503 - lr: 0.0100
Epoch 49/256
85/85 [==============================] - 1s 6ms/step - loss: 0.6748 - accuracy:
0.7878 - val_loss: 0.3278 - val_accuracy: 0.8844 - lr: 0.0100
Epoch 50/256
85/85 [==============================] - 1s 6ms/step - loss: 0.5874 - accuracy:
0.8145 - val_loss: 0.4199 - val_accuracy: 0.8571 - lr: 0.0100
Epoch 51/256
85/85 [==============================] - 1s 6ms/step - loss: 0.5987 - accuracy:
0.8160 - val_loss: 0.4945 - val_accuracy: 0.8367 - lr: 0.0100
Epoch 52/256
85/85 [==============================] - 1s 6ms/step - loss: 0.6316 - accuracy:
0.7849 - val_loss: 0.3030 - val_accuracy: 0.8980 - lr: 0.0100
Epoch 53/256
84/85 [============================>.] - ETA: 0s - loss: 0.5788 - accuracy:
0.8213
Epoch 53: ReduceLROnPlateau reducing learning rate to 0.004999999888241291.
85/85 [==============================] - 0s 6ms/step - loss: 0.5757 - accuracy:
0.8234 - val_loss: 0.3689 - val_accuracy: 0.8435 - lr: 0.0100
Epoch 54/256
85/85 [==============================] - 1s 6ms/step - loss: 0.4544 - accuracy:
0.8605 - val_loss: 0.3829 - val_accuracy: 0.8435 - lr: 0.0050
Epoch 55/256
85/85 [==============================] - 1s 6ms/step - loss: 0.4238 - accuracy:
0.8605 - val_loss: 0.3869 - val_accuracy: 0.8435 - lr: 0.0050
Epoch 56/256
85/85 [==============================] - 1s 6ms/step - loss: 0.4801 - accuracy:
0.8561 - val_loss: 0.2324 - val_accuracy: 0.9252 - lr: 0.0050
Epoch 57/256
85/85 [==============================] - 1s 6ms/step - loss: 0.4333 - accuracy:
0.8769 - val_loss: 0.2615 - val_accuracy: 0.9116 - lr: 0.0050
Epoch 58/256
85/85 [==============================] - 1s 6ms/step - loss: 0.4168 - accuracy:
0.8635 - val_loss: 0.2479 - val_accuracy: 0.9252 - lr: 0.0050
Epoch 59/256
85/85 [==============================] - 0s 6ms/step - loss: 0.3801 - accuracy:
0.8813 - val_loss: 0.2569 - val_accuracy: 0.9116 - lr: 0.0050
Epoch 60/256
85/85 [==============================] - 0s 6ms/step - loss: 0.3095 - accuracy:
0.9065 - val_loss: 0.2950 - val_accuracy: 0.8980 - lr: 0.0050
```

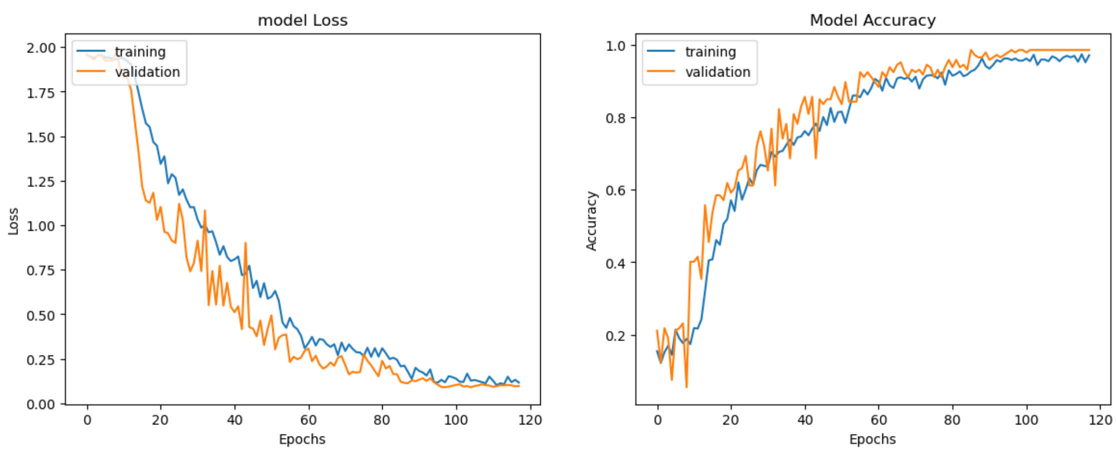
```
Epoch 61/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3381 - accuracy:
0.8991 - val_loss: 0.3079 - val_accuracy: 0.8844 - lr: 0.0050
Epoch 62/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3734 - accuracy:
0.8739 - val_loss: 0.2367 - val_accuracy: 0.9252 - lr: 0.0050
Epoch 63/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3251 - accuracy:
0.9095 - val_loss: 0.2677 - val_accuracy: 0.9116 - lr: 0.0050
Epoch 64/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3612 - accuracy:
0.8887 - val_loss: 0.2189 - val_accuracy: 0.9388 - lr: 0.0050
Epoch 65/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3572 - accuracy:
0.8813 - val_loss: 0.1954 - val_accuracy: 0.9252 - lr: 0.0050
Epoch 66/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3316 - accuracy:
0.9080 - val_loss: 0.2074 - val_accuracy: 0.9456 - lr: 0.0050
Epoch 67/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3169 - accuracy:
0.9110 - val_loss: 0.2301 - val_accuracy: 0.9524 - lr: 0.0050
Epoch 68/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3321 - accuracy:
0.9065 - val_loss: 0.2108 - val_accuracy: 0.9252 - lr: 0.0050
Epoch 69/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2701 - accuracy:
0.9110 - val_loss: 0.2559 - val_accuracy: 0.9116 - lr: 0.0050
Epoch 70/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3420 - accuracy:
0.8991 - val_loss: 0.2651 - val_accuracy: 0.9320 - lr: 0.0050
Epoch 71/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2940 - accuracy:
0.9125 - val_loss: 0.2142 - val_accuracy: 0.9252 - lr: 0.0050
Epoch 72/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3313 - accuracy:
0.8798 - val_loss: 0.1626 - val_accuracy: 0.9320 - lr: 0.0050
Epoch 73/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3060 - accuracy:
0.9050 - val_loss: 0.1776 - val_accuracy: 0.9184 - lr: 0.0050
Epoch 74/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2877 - accuracy:
0.9154 - val_loss: 0.1732 - val_accuracy: 0.9456 - lr: 0.0050
Epoch 75/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2852 - accuracy:
0.9169 - val_loss: 0.1757 - val_accuracy: 0.9388 - lr: 0.0050
Epoch 76/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2659 - accuracy:
0.9154 - val_loss: 0.2740 - val_accuracy: 0.9116 - lr: 0.0050
```

```
Epoch 77/256
85/85 [==============================] - 0s 6ms/step - loss: 0.3126 - accuracy:
0.9080 - val_loss: 0.2393 - val_accuracy: 0.9320 - lr: 0.0050
Epoch 78/256
85/85 [==============================] - 0s 6ms/step - loss: 0.2614 - accuracy:
0.9243 - val_loss: 0.2141 - val_accuracy: 0.9116 - lr: 0.0050
Epoch 79/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3101 - accuracy:
0.8902 - val_loss: 0.1818 - val_accuracy: 0.9388 - lr: 0.0050
Epoch 80/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2632 - accuracy:
0.9303 - val_loss: 0.1523 - val_accuracy: 0.9592 - lr: 0.0050
Epoch 81/256
85/85 [==============================] - 1s 6ms/step - loss: 0.3096 - accuracy:
0.9154 - val_loss: 0.2389 - val_accuracy: 0.9388 - lr: 0.0050
Epoch 82/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2810 - accuracy:
0.9199 - val_loss: 0.1957 - val_accuracy: 0.9592 - lr: 0.0050
Epoch 83/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2491 - accuracy:
0.9273 - val_loss: 0.2101 - val_accuracy: 0.9388 - lr: 0.0050
Epoch 84/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2563 - accuracy:
0.9139 - val_loss: 0.1642 - val_accuracy: 0.9456 - lr: 0.0050
Epoch 85/256
83/85 [============================>.] - ETA: 0s - loss: 0.2453 - accuracy:
0.9195
Epoch 85: ReduceLROnPlateau reducing learning rate to 0.0024999999441206455.
85/85 [==============================] - 1s 6ms/step - loss: 0.2450 - accuracy:
0.9184 - val_loss: 0.1638 - val_accuracy: 0.9320 - lr: 0.0050
Epoch 86/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2081 - accuracy:
0.9273 - val_loss: 0.1216 - val_accuracy: 0.9864 - lr: 0.0025
Epoch 87/256
85/85 [==============================] - 1s 6ms/step - loss: 0.2118 - accuracy:
0.9318 - val_loss: 0.1153 - val_accuracy: 0.9728 - lr: 0.0025
Epoch 88/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1760 - accuracy:
0.9436 - val_loss: 0.1134 - val_accuracy: 0.9660 - lr: 0.0025
Epoch 89/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1378 - accuracy:
0.9629 - val_loss: 0.1294 - val_accuracy: 0.9660 - lr: 0.0025
Epoch 90/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1998 - accuracy:
0.9421 - val_loss: 0.1243 - val_accuracy: 0.9796 - lr: 0.0025
Epoch 91/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1821 - accuracy:
0.9347 - val_loss: 0.1331 - val_accuracy: 0.9592 - lr: 0.0025
```

```
Epoch 92/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1735 - accuracy:
0.9451 - val_loss: 0.1410 - val_accuracy: 0.9660 - lr: 0.0025
Epoch 93/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1563 - accuracy:
0.9585 - val_loss: 0.1264 - val_accuracy: 0.9728 - lr: 0.0025
Epoch 94/256
77/85 [==========================>…] - ETA: 0s - loss: 0.2018 - accuracy:
0.9508
Epoch 94: ReduceLROnPlateau reducing learning rate to 0.0012499999720603228.
85/85 [==============================] - 1s 6ms/step - loss: 0.1910 - accuracy:
0.9540 - val_loss: 0.1414 - val_accuracy: 0.9660 - lr: 0.0025
Epoch 95/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1197 - accuracy:
0.9629 - val_loss: 0.1192 - val_accuracy: 0.9728 - lr: 0.0012
Epoch 96/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1159 - accuracy:
0.9629 - val_loss: 0.1057 - val_accuracy: 0.9796 - lr: 0.0012
Epoch 97/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1324 - accuracy:
0.9585 - val_loss: 0.0922 - val_accuracy: 0.9864 - lr: 0.0012
Epoch 98/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1186 - accuracy:
0.9629 - val_loss: 0.0908 - val_accuracy: 0.9796 - lr: 0.0012
Epoch 99/256
85/85 [==============================] - ETA: 0s - loss: 0.1525 - accuracy:
0.9570
Epoch 99: ReduceLROnPlateau reducing learning rate to 0.0006249999860301614.
85/85 [==============================] - 1s 6ms/step - loss: 0.1525 - accuracy:
0.9570 - val_loss: 0.0938 - val_accuracy: 0.9864 - lr: 0.0012
Epoch 100/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1474 - accuracy:
0.9570 - val_loss: 0.0990 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 101/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1387 - accuracy:
0.9629 - val_loss: 0.1035 - val_accuracy: 0.9796 - lr: 6.2500e-04
Epoch 102/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1216 - accuracy:
0.9555 - val_loss: 0.1074 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 103/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1205 - accuracy:
0.9733 - val_loss: 0.0947 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 104/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1671 - accuracy:
0.9451 - val_loss: 0.0976 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 105/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1273 - accuracy:
0.9599 - val_loss: 0.0904 - val_accuracy: 0.9864 - lr: 6.2500e-04
```

```
Epoch 106/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1314 - accuracy:
0.9599 - val_loss: 0.0974 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 107/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1255 - accuracy:
0.9555 - val_loss: 0.1010 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 108/256
82/85 [===========================>..] - ETA: 0s - loss: 0.1210 - accuracy:
0.9677
Epoch 108: ReduceLROnPlateau reducing learning rate to 0.0003124999930150807.
85/85 [==============================] - 1s 6ms/step - loss: 0.1195 - accuracy:
0.9688 - val_loss: 0.1065 - val_accuracy: 0.9864 - lr: 6.2500e-04
Epoch 109/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1125 - accuracy:
0.9644 - val_loss: 0.1025 - val_accuracy: 0.9864 - lr: 3.1250e-04
Epoch 110/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1504 - accuracy:
0.9555 - val_loss: 0.1001 - val_accuracy: 0.9864 - lr: 3.1250e-04
Epoch 111/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1277 - accuracy:
0.9659 - val_loss: 0.0931 - val_accuracy: 0.9864 - lr: 3.1250e-04
Epoch 112/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1019 - accuracy:
0.9703 - val_loss: 0.0968 - val_accuracy: 0.9864 - lr: 3.1250e-04
Epoch 113/256
82/85 [===========================>..] - ETA: 0s - loss: 0.1148 - accuracy:
0.9649
Epoch 113: ReduceLROnPlateau reducing learning rate to 0.00015624999650754035.
85/85 [==============================] - 1s 6ms/step - loss: 0.1130 - accuracy:
0.9662 - val_loss: 0.1021 - val_accuracy: 0.9864 - lr: 3.1250e-04
Epoch 114/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1071 - accuracy:
0.9703 - val_loss: 0.1015 - val_accuracy: 0.9864 - lr: 1.5625e-04
Epoch 115/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1500 - accuracy:
0.9540 - val_loss: 0.1036 - val_accuracy: 0.9864 - lr: 1.5625e-04
Epoch 116/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1191 - accuracy:
0.9748 - val_loss: 0.1013 - val_accuracy: 0.9864 - lr: 1.5625e-04
Epoch 117/256
85/85 [==============================] - 1s 6ms/step - loss: 0.1325 - accuracy:
0.9525 - val_loss: 0.0959 - val_accuracy: 0.9864 - lr: 1.5625e-04
Epoch 118/256
82/85 [===========================>..] - ETA: 0s - loss: 0.1181 - accuracy:
0.9708Restoring model weights from the end of the best epoch: 98.
85/85 [==============================] - 1s 6ms/step - loss: 0.1172 - accuracy:
0.9718 - val_loss: 0.0975 - val_accuracy: 0.9864 - lr: 1.5625e-04
Epoch 118: early stopping
```

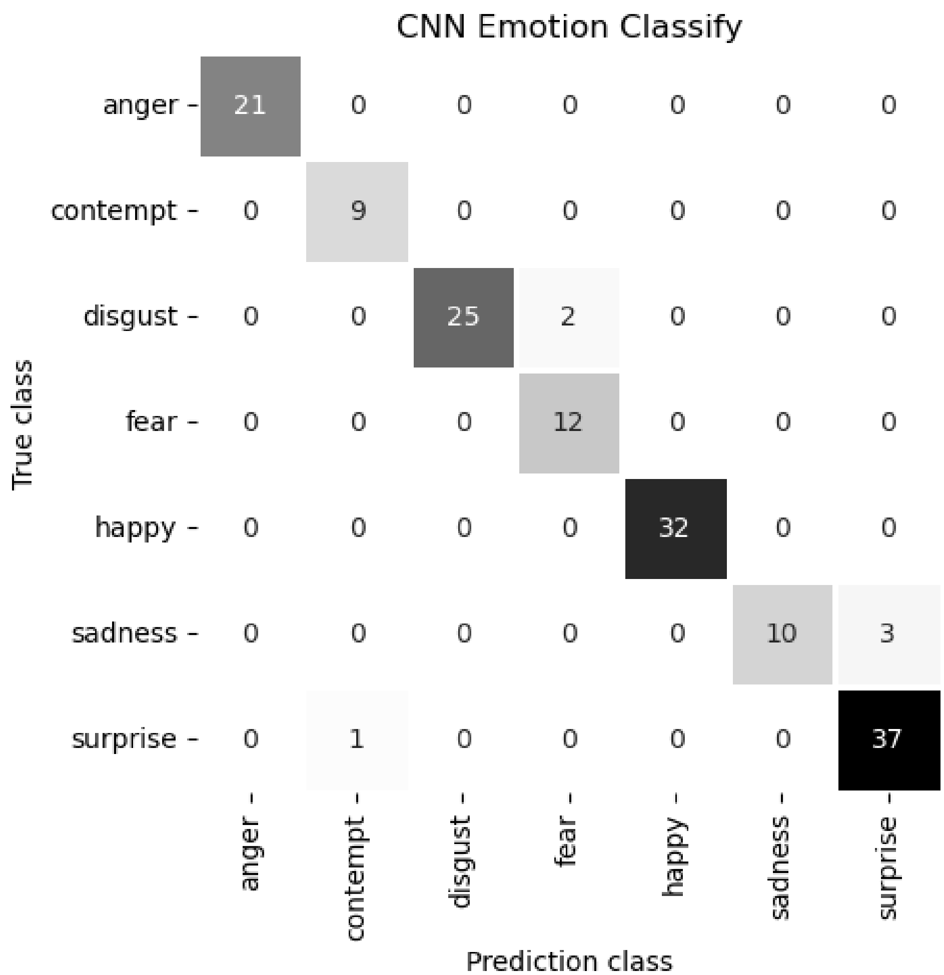
[407]:
```
plt.figure(figsize=(14,5))
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['training', 'validation'], loc='upper left')

plt.subplot(1,2,1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['training', 'validation'], loc='upper left')
plt.show()
```



[408]:
```
train_loss, train_accu = model.evaluate(trainingData)
val_loss, val_accu = model.evaluate(validationData)
test_loss, test_accu = model.evaluate(testingData)

print("final train accuracy = {:.2f} , validation accuracy = {:.2f}".
↪format(train_accu*100, val_accu*100))

from sklearn.metrics import confusion_matrix
import pandas as pd
import numpy as np
```

```
Y_pred = model.predict(testingData)
y_pred = np.argmax(Y_pred, axis=1)

label = ['anger','contempt','disgust','fear','happy','sadness','surprise']
labels = {0 : 'anger', 1 : 'contempt', 2 : 'disgust', 3 : 'fear', 4 : 'happy',5␣
 ↪:'sadness',6 :'surprise'}



#Transform to df for easier plotting
cm = confusion_matrix(testingData.classes, y_pred)
cm_df = pd.DataFrame(cm, index = label,
                     columns = label
                    )

import seaborn as sns

plt.figure(figsize = (5,5))
sns.heatmap(cm_df, annot = True,cmap='Greys',cbar=False,linewidth=2,fmt='d')
plt.title('CNN Emotion Classify')
plt.ylabel('True class')
plt.xlabel('Prediction class')
plt.show()
```

```
86/86 [==============================] - 0s 5ms/step - loss: 0.0932 - accuracy:
0.9707
19/19 [==============================] - 0s 2ms/step - loss: 0.0908 - accuracy:
0.9796
19/19 [==============================] - 0s 2ms/step - loss: 0.1938 - accuracy:
0.9605
final train accuracy = 97.07 , validation accuracy = 97.96
19/19 [==============================] - 0s 2ms/step
```



[422]:
```
from keras.utils import load_img, img_to_array
from keras import models
model2 = models.load_model('FER1.keras')

def choose_image_and_predict(image):
    img  = cv2.imread(image, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (48, 48))
    img = img/255
    img = np.expand_dims(img, axis=0)
    img = np.stack([img] * 3, axis=-1)
    pred = model.predict(img)
    label=np.argmax(pred,axis=1)[0]
    return labels[label]


fig = plt.figure(figsize=(10, 7))
```

```
rows = 2
columns = 4

fig.add_subplot(rows, columns, 1)
plt.imshow(load_img("happy.jpg"))
plt.axis('off')
plt.title(choose_image_and_predict("happy.jpg"))

fig.add_subplot(rows, columns, 2)
plt.imshow(load_img("sad.png"))
plt.axis('off')
plt.title(choose_image_and_predict("sad.png"))

fig.add_subplot(rows, columns, 3)
plt.imshow(load_img("fear2.png"))
plt.axis('off')
plt.title(choose_image_and_predict("fear2.png"))


fig.add_subplot(rows, columns, 4)
plt.imshow(load_img("surprise.png"))
plt.axis('off')
plt.title(choose_image_and_predict("surprise.png"))


fig.add_subplot(rows, columns, 5)
plt.imshow(load_img("contempt.png"))
plt.axis('off')
plt.title(choose_image_and_predict("contempt.png"))

fig.add_subplot(rows, columns, 6)
plt.imshow(load_img("disgust.png"))
plt.axis('off')
plt.title(choose_image_and_predict("disgust.png"))

fig.add_subplot(rows, columns, 7)
plt.imshow(load_img("anger.png"))
plt.axis('off')
plt.title(choose_image_and_predict("anger.png"))
```

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 13ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
```

```
1/1 [==============================] - 0s 14ms/step
```

[422]:
```
Text(0.5, 1.0, 'anger')
```